Key Knowledge Delivery Factors affecting Software Development Productivity

by

Zheng Sun

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Master of Engineering in Telecommunications Technology Management

Department of Systems and Computer Engineering

Carleton University

Ottawa, Canada, K1S 5B6

May, 2004

© Copyright 2004 Zheng Sun

Library and
Archives Canada

Bibliothèque et
Archives Canada

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-612-97467-7

Our file Notre référence

ISBN: 0-612-97467-7

The author has granted a non-exclusive license allowing the Library and Archives Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

PAGINATION ERROR.

TEXT COMPLETE.

ERREUR DE PAGINATION.

LE TEXTE EST COMPLET.

ABSTRACT

Software development (SWD) is a human and knowledge intensive activity. Many software development organizations apply Knowledge Management System (KMS) to systematically create, gather, store and distribute their in-house knowledge. Yet, employing KMS might introduce some risks as well as benefits to organizations. Better understanding of how different characteristics of KMS affect software development productivity will tremendously help organizations increase the benefits brought by the introduction of KMS, and reduce its potential risks at the same time.

Knowledge delivery, or knowledge dissemination, is a core part of KM systems. Based on literature review, this thesis identifies four key factors in knowledge delivery: a) pull or push approach for knowledge delivery, b) time to deliver knowledge, c) degree to which knowledge delivery is embedded in development processes, d) the form of knowledge delivery. The author builds a research model and four hypotheses anchored around this model, on how knowledge delivery factors affect software development productivity. Data used to test the research model and hypotheses was collected from questionnaires completed by 41 software development companies headquartered in North America. Research results suggest that: 1) software development will be more efficient if pull approach is used for knowledge delivery; 2) just-in-time knowledge delivery (knowledge is delivered close to the time such knowledge is needed) will increase software development productivity.

ACKNOWLEDGEMENT

I am very thankful to my thesis director Dr. Samuel A. Ajila for his supervision and suggestions on many drafts of this thesis, and for his patience, guidance, and unwavering support.

I thank Dr. Tony Bailetti and Dr. John Callahan for their support and suggestions for my research.

I highly appreciate the time and efforts taken by all respondents in answering the questionnaires for this research.

At last, I sincerely thank my lovely wife, Xiaohan, and my parents for their continuing support and understanding throughout the duration of this research.

TABLE OF CONTENT

1	INTRODUCTION.....	9
1.1	Background.....	9
1.2	Research Objective	10
1.3	Research Motivations.....	12
1.4	Contributions.....	13
2	LITERATURE REVIEW	16
2.1	Knowledge & Knowledge Management.....	16
2.2	Knowledge Management in Software Engineering	18
2.3	Knowledge Management in New Product Development.....	21
3	HYPOTHESIS AND PROPOSITIONS.....	24
3.1	How knowledge is delivered to receivers	24
3.2	When is the best time to deliver knowledge	26
3.3	How knowledge delivery is embedded in business (or development) process.	27
3.4	Form of knowledge delivery: Inter-personal communication or reading document.....	29
4	RESEARCH DESIGN	32
4.1	Research Method	32
4.1.1	Unit of Analysis	32
4.1.2	Sample selection	32
4.1.3	Questionnaire	34
4.2	Variables and Measures	35

4.2.1	Independent Variables	35
4.2.2	Dependent Variables	36
4.2.3	Control Variables	38
4.2.3.1	Organization size	38
4.2.3.2	Knowledge Worker's Experience	38
4.3	Research Model	39
5	RESEARCH RESULTS	40
5.1	Samples	40
5.2	Data analysis	42
5.3	Hypothesis Testing.....	43
5.3.1	H1, Knowledge Delivery Approach	43
5.3.2	H2, Time of Knowledge Delivery	45
5.3.3	H3, Embedded-ness of Knowledge Delivery in SWD Process	49
5.3.4	H4, Knowledge Delivery Form.....	51
5.4	Summary	53
6	DISCUSSION OF RESULTS	55
6.1	Influence of Control Variables on Knowledge Delivery	55
6.1.1	Organizational Size	55
6.1.2	Influence of Knowledge Worker's Experience.....	57
6.2	Result Analysis for Knowledge Delivery Factors	59
7	CONCLUSIONS, LIMITATIONS AND FUTURE WORK	63
7.1	Conclusions.....	63
7.2	Limitations	64

7.3	Suggestions for future research.....	65
8	REFERENCES.....	68
APPENDIX A	Letter of Information.....	76
APPENDIX B	Questionnaire	79

LIST OF FIGURES

Figure 1. Knowledge Management As Cyclic Process.....	11
Figure 2. Experience Factory [44]	19
Figure 3. Research Model.....	39
Figure 4. Box plot of SWD Productivity by Knowledge Delivery Approaches.....	44
Figure 5. Measurement of Knowledge Delivery Time	46
Figure 6. Box plot of SWD Productivity by Knowledge Delivery Forms.....	52

LIST OF TABLES

Table 1. Descriptive Statistics (Sample Size = 41).....	41
Table 2. Pearson Correlations Coefficients for Study Variables	41
Table 3. Interactions between Control Variables and Delivery Approaches.....	44
Table 4. Two Independent Samples t Test Results of H1	45
Table 5. Interactions between Control Variables and Independent Variables for H2	47
Table 6. Descriptive Statistics of T_{ab}	48
Table 7. Spearman correlation of T_{ab} and SWD productivity	48
Table 8. Results of Quadratic Regression Analysis for H2	48
Table 9. Interactions between Control Variables and Independent Variables for H3	50
Table 10. Interactions between Control Variables and Delivery Forms.....	52
Table 11. Two Independent Samples t Test Results of H4.....	53
Table 12. Summary of Hypothesis Testing.....	54
Table 13. Chi-Square Test Result for Organizational Size.....	56
Table 14. Two Independent Samples T Test Results for Organizational Size	57
Table 15. Chi-Square Test Result for Experience	58
Table 16. Two Independent Samples T Test Results for Experience	59

1 INTRODUCTION

1.1 Background

Software development, like other New Product Development (NPD) practices, can be considered as a problem-solving practice that is, building software to solve the domain problems of end-users. It is a human and knowledge intensive activity. The main asset of a software development organization is its knowledge repository, or otherwise called *intellectual capital*. The success of software development depends to a large extent on how the organization locates and utilizes its in-house knowledge.

The ability to better access and utilize their in-house intellectual capital will provide organizations the necessary expertise needed for making timely and better decisions, resolving problems, and taking proactive actions. It can take advantage of best practices, eliminate redundant efforts and speed products to market, thus saving company unnecessary development costs, improving development team's productivity and increasing the quality of product development as well.

Facing great challenges in managing their in-house knowledge, many software development organizations apply different types of Knowledge Management System (KMS) to address these challenges. Based on the state-of-art information and communication technologies, KMS promises to provide organizations a systematic and organized way to create, gather, store and distribute knowledge and help people

distributed all over the world to collaborate with each others [42]. It involves efficiently connecting those who know with those who need to know, and converting personal knowledge into organizational knowledge [58].

Some of the major motivations for applying KMS in software development are [48]: reducing time to market, decreasing development cost, increasing quality, collaboration, and knowledge sharing. It is assumed by software industries that applying KMS to software development will necessarily have positive effects on the development process, which will in turn lead to improved software development performance. However, recent studies show that applying KM systems/tools could bring risks as well as benefits to NPD organizations [16]. Therefore, to provide better KM system to software development organizations, it is necessary to investigate what “better” means, what are desirable characteristics of KMS that will increase software development performance improvement, as well as reduce the risks brought by KMS. This leads to the main focus of this thesis: how different characteristics, or variables, in KMS affect software development performance.

1.2 Research Objective

According to Fischer and Ostwald [21], from design perspective, Knowledge Management is actually a cyclic process involving three related activities: creation, integration and dissemination. As illustrated in Figure 1, knowledge workers constantly create knowledge as they work, and then the created knowledge is formalized and

integrated into knowledge repositories or so-called organization memories, so that it can be shared by the whole organization. Yet, knowledge saved in organization memories is useless until it is delivered to the person who needs such knowledge and applies such knowledge in the real working environment to solve practical problems.

Figure 1. Knowledge Management As Cyclic Process

In this cyclic three-phase KM model, each phase plays an irreplaceable role in the whole KM cycle. As computer-aided KM system is more commonly used in the domain of knowledge distribution [19], this research focuses on the knowledge dissemination, or the knowledge delivery phase. The objective of this research is to investigate how knowledge is delivered, shared in software development organizations, and explore the effect of different factors of knowledge delivery on software development productivity.

It is worth nothing that KMS can only deliver knowledge to potential users' desktop, but it cannot dictate what they do with it thereafter [19]. The delivered knowledge is useless unless it is applied to software development activities. This study

investigates the effect of knowledge delivery factors on software development productivity based on the assumption that once the knowledge is delivered to the receivers, it will be applied in their work.

1.3 Research Motivations

A lot of the knowledge management research works found in the literature deal with specific elements of knowledge enterprise in general, such as knowledge transfer [55], organization memory [17], knowledge strategy [60], design of KM models [36] [53] [2]. There also exist various research models, mostly reported by case studies, addressing KM challenges in software development [8] [27] [28]. However, there is a lack of research on how knowledge delivery, which is the core part of computer-aided KMS, affects performance of New Product Development, especially of software development. Therefore, the major goal of this research is to fill such research gap by identifying key knowledge delivery factors in software development, and quantitatively analyzing their impacts on software development productivity.

In particular, this research is largely motivated by Cooper's paper [16] presented in a recent issue of Engineering and Technology Management Journal. In the paper, Cooper reviewed various KMS and collaboration tools (CT) to help New Product Development (NPD) teams acquire, develop, and manage their knowledge. She discovered that applying KMS/CT would bring risks as well as benefits to NPD organizations. Even worse, some NPD organizations were suffering frustrations

introduced by KMS/CT, which (may) remarkably decrease the KM system's positive effect on improving NPD performance. She divided the frustrations encountered by NPD organizations from using KMS/CT into three categories: a) disrupting the natural flow of activity. b) altering roles i.e. where the use of the tool substantially changes the roles of team members. c) lack of context - where the tool hides or eliminates important contextual information. Based on that, she identified a number of requirements that KMS/CT should meet to support knowledge sharing for innovation. Finally, she presented a set of research challenges whose results would address design and use issues of KMS/CT applied in NPD organizations.

Inspired by one of the research challenges identified in Cooper's paper on the use of KMS/CT in New Product Development, this study is going to investigate how different characteristics, or factors, in the knowledge delivery phase of KM life cycle impact new product development performance. More specifically, the research is focused on the software development activities, as software development is a knowledge-extensive activity, and it is one of the most important categories of new product development.

1.4 Contributions

This thesis makes four contributions.

First of all, this thesis identifies four major factors of knowledge delivery in software development.

Secondly, most research models in knowledge management literature are focused on how knowledge is acquired and saved in knowledge repositories, or so-called organization memories. There is a lack of research on how knowledge is delivered to end users in new product development teams, especially in software development teams. This research is the first empirical study on how knowledge is delivered in the software development, as well as on the impact of key knowledge delivery factors on software development productivity.

Thirdly, in current research literature on software engineering, research studies on the effect of KM models in software development are mostly conducted through case studies [8] [27] [28], which targeted a small number of software development organizations. This research work provides a research model that quantitatively analyse the impact of knowledge delivery in software development organizations based on a much wider population of samples (data are collected from 41 software development companies).

Lastly, this research presents a number of characteristics that knowledge delivery system should possess to better support software development activities. Such results can help vendors of KM system or tools to build better KM products more suitable for software development organization. Furthermore, managers in software development

organizations can use the results of this research as a guide for applying KMS in their organizations, in order to obtain more significant improvement in software development performance from introducing KM systems.

The rest of this thesis is structured as follows. Chapter 2 provides a literature review on Knowledge Management and its application in Software Engineering and New Product Development. Chapter 3 develops four testable hypotheses on how key knowledge delivery factors affect software development productivity. Chapter 4 describes research methods, variables and the measurements used to test the hypotheses. Chapter 5 presents research results and analysis of the collected data. Chapter 6 discusses the research results. Finally, chapter 7 provides the conclusions and limitations of this research, as well as the suggestions for future studies.

2 LITERATURE REVIEW

2.1 Knowledge & Knowledge Management

According to the Webster dictionary [39], Knowledge is “the fact or condition of knowing something with familiarity gained through experience or association; acquaintance with or understanding of a science, art, or technique; the fact or condition of being aware of something”. In software organizations, knowledge is “embedded not only in documents or repositories, but also in organizational routines, processes, practices, and norms” [19].

In general, knowledge can be categorized as explicit knowledge or tacit knowledge [41]. Explicit knowledge, also known as codified knowledge, corresponds to the information and skills that can be formally articulated, so that it can be easily transferred or shared. Tacit knowledge is highly personal and hard to formalize. One of the most important characteristics of knowledge is that it has several levels of refinements [48]: *Data* consists of discrete, objective facts about events. *Information* is organized data for decision making. Broader than data and information, *Knowledge* requires understanding of information. And *Experience* is applied knowledge.

Knowledge Management (KM) is an emerging discipline dealing with how to manage and capitalize on an organization’s internal intellectual capital. It has many definitions [42] [58] [20], however, the major goal of KM is [56]: “to make the enterprise

act as intelligently as possible to secure its viability and overall success". As described by Wiig [57], the essential aspects of Knowledge Management are

- Survey, develop, maintain and secure the intellectual and knowledge resources of the enterprise.
- Determine the knowledge and expertise required to perform work tasks, organise it, make the requisite knowledge available, package it, and distribute it to the relevant points of action.
- Provide knowledge architecture so that the enterprise's facilities, procedures, guidelines, standards, examples, and practices facilitate and support active Knowledge Management as part of the organisation's practices and culture.

In the current literature, research works related to knowledge management in software development can be divided into two streams based on different perspectives: *Knowledge Management Software Engineering* and *Knowledge Management in New Product Development*. Remaining part of this chapter provides a detailed review of these two research streams.

2.2 Knowledge Management in Software Engineering

From the perspective of software engineering, software development involves various knowledge-intensive activities: elicitation of customer's requirement, identification of software development architecture, collection of experience about project management, building development processes, etc. Rus and Lindvall [48] reviewed five types of needs for KM in software development:

- Acquiring knowledge about new technologies;
- Accessing domain knowledge;
- Sharing knowledge about local policies and practices;
- Capturing knowledge and knowing who knows what;
- Collaborating and sharing knowledge.

Especially, one discipline of software engineering: *Experimental Software Engineering* (ESE) [47] puts particular focus on knowledge management in software development. ESE addresses problems in software development via systematically designed experiments and other empirical studies. It is built on continuous learning and systematic reuse of experience, which is defined through the Quality Improvement Paradigm (QIP) [38].

A framework to implement QIP is the *Experience Factory* [7]. In such framework, experience is collected from software development projects, and is packaged

and stored in an experience base. It supports software project development by analyzing and synthesizing all kinds of experience, working as a repository for such experience, and supplying that experience to different projects. It will help new software development projects with earlier experiences. Moreover, it can also suggest improvements in processes based on collected experiences.

Figure 2. Experience Factory [44]

Many tools and technologies have been designed to support KM in software engineering, like Experience Management Systems [53], MILOS project [36]. Some of them [5] [6] are based on Case-Based Reasoning (CBR) [1] to preserve and retrieve

experiences. Some apply advanced probabilistic indexing techniques, and others combine manual and automatic generation of taxonomies.

By and large, knowledge-based tools and research prototypes for software engineering are focused on specific phase of software development cycle. For instance, tools like *Kibitzer* [49], and *RFML Workbench* [24] focus on requirement elicitation; *CAESAR* [22], *RT-Syn* [52], and *Comet* [35] are used for system design; *KIDS* [51], *CodeBroker* [59] and *CodeFiner* [25] are used for code generation. In addition, many software development projects apply predictive models like *COCOMO* [10] and *COCOTS* [4] to guide future project development based on prior experiences.

Researchers in the software engineering literature present various reports on the effect of knowledge management on organizations as well. Most of them are case studies, such as implementation of Experience Factory in NASA Software Engineering Laboratory [8] and Daimler Chrysler [27] [31]; an experience-based effort estimation and risk management system used in Telenor Telecom Software [29]; “Experience Broker” and “Experience communicators” model deployed in Ericsson Software Technology [28]; “Engineering Process Improvement Framework” deployed in ICL High Performance Systems in the UK [13]; and “Knowledge Broker” system deployed in SD&M in Germany [11].

2.3 Knowledge Management in New Product Development

The review of research literature on KM in New Product Development is based on five recently published journal articles: Mohrman et al. [40], Cummings and Teng [18], Cooper [16], Leenders et al. [32] and O'Sullivan [43].

Mohrman et al. [40] presented a conceptual model of the NPD organization as a knowledge enterprise, which includes four parts: (1) contextual organizational elements; (2) knowledge work behaviours; (3) knowledge outcomes; and (4) effectiveness. They quantitatively evaluated how different factors in contextual organizational elements and knowledge work behaviours affect organization's capability of acquiring new knowledge and its effective use. They believed that knowledge and knowing capabilities of the firm will translate into NPD firm effectiveness in the marketplace, and it is very important to adopt systemic processes to guide the work.

Cummings and Teng [18] focused on the knowledge transfer between R&D organizations. They defined transfer success as commitment, satisfaction and ownership by recipients of the transferred knowledge. They found that knowledge that is embedded in a source organization's routines and practices transfers less successfully than non-embedded knowledge, and more articulated (coded) knowledge transfers less successfully than less articulated knowledge. They argued that knowledge transfer success is associated with several key variables, including: a) both R&D units' understanding where the desired knowledge resides within the source. b) extent to which the parties share

similar knowledge bases, and the extent of interactions between the source and the recipient. c) transfer the knowledge and d) participate in an articulation process through which the source's knowledge is made accessible to the recipient.

Cooper [16] described a number of problems that NPD teams have experienced in using KMS for innovation. Her paper is the major source of inspiration for this research. Please refer to section 1.3 for more detailed review of her work.

Leenders et al. [32] investigated knowledge transfer development and sharing in NPD teams. They examined the effect of frequency of communication and communication centralization on NPD creativity. They argued that creativity of NPD team is impeded by both a low and high density of the communication network and by centralization of the communication network. In addition, they found that the creativity of NPD teams decreases as the longevity of team membership increases. Their research suggested that knowledge-sharing in NPD organization should not be continuous, nor should the team leader be the heavyweight centralized repository for knowledge if creativity is the objective.

O'Sullivan [43] studied work patterns in a virtual multilateral (multi-organization, multi-team) development organization (VMDO) composed of a lead firm and its suppliers. He found that standardization of working contextual information imposed by leading company and synchronization of work-flow ease the communications between collaborative partners, constitute a context for effective virtual collaboration. He also

believed that modularization of product-architecture should also be associated with modularization of organizational form.

3 HYPOTHESIS AND PROPOSITIONS

Up to now, researchers in the literature of KM and organization learning have focused on four aspects of knowledge delivery based on different perspectives:

- which approach is better to deliver knowledge: pull or push
- when is the best time to deliver knowledge
- how knowledge delivery is embedded in business (or software development) process
- which form is better for knowledge delivery: inter-personal communication or reading documents

This chapter identifies four key knowledge delivery factors in software development by reviewing major research streams. In addition, four hypotheses are proposed to test the impact of these factors on software development productivity.

3.1 How knowledge is delivered to receivers

Reported in the literature, there have been two ways of delivering knowledge: ‘pull’ approach and ‘push’ approach. The former refers to technologies that aim at pulling knowledge from existing knowledge repositories or organization memories to knowledge users. In the pull approach, users are expected to search for knowledge by themselves. On the other hand, the ‘push’ approach aims at providing knowledge to users without their

prior interactions. Electronic newsletters, traditional class-room learning or training are examples of the push approach of knowledge delivery [34].

“Push” approach of knowledge delivery separates learning from working. When knowledge is pushed to users, it is assumed that the knowledge being delivered is useful for knowledge worker and it is more likely to be reused in the future. Yet, in real working environments, this assumption is problematic, because it is very difficult to forecast which knowledge is more useful in the future. Furthermore, this approach pays little consideration to recipients’ special needs and working context in their working environment. By broadcasting de-contextualized information to end users, the push approach can easily cause information overload.

In contrast, by using pull approach, knowledge workers already know what kind of knowledge they will need for their work, and they take initiatives to search and learn such knowledge. In that approach, there’s no gap between learning and working. It is a learning-on-demand approach, as what is learnt is what they need. Above considerations lead to the first hypothesis:

Hypothesis 1: *Software development is more productive if majority of knowledge is delivered by pull approach rather than push approach.*

3.2 When is the best time to deliver knowledge

According to Kevin Cole [14], successful knowledge delivery should provide users with the knowledge required by the task they are performing at the time they are actually performing the task. He argued that knowledge delivery should take place “soon enough that it is applied to the appropriate situation, and late enough that the user does not have to go through training or information overload”.

A good example of such “Just-in-Time” knowledge delivery is Microsoft Office help system. With the help of such system, users do not necessarily understand all functions of the software they are using. Instead, they can depend on the software’s sophisticated help system to guide them to the functions they need, as well as the detailed context information on how to use these functions.

As far as software development is concerned, most types of knowledge applied in software development are under fast change. Generally, knowledge required in software development should be the latest, most “fresh” knowledge rather than outdated one. If knowledge is delivered to software engineers too early before the time they need it, the delivered knowledge might have turned obsolete and is not useful at all by the time such knowledge is needed. On the other hand, software engineers expect knowledge to be delivered not too late after the time they need it, as they probably don’t have time to waste on waiting for or looking for the knowledge. These considerations lead to the second hypothesis:

***Hypothesis 2:** Software development productivity is impacted positively if knowledge required is delivered close to the time such knowledge is needed.*

3.3 How knowledge delivery is embedded in business (or development) process

Studies show that people are often not aware of information that might be relevant to them, even though this knowledge has been explicitly stored in the company's organizational memory [33]. Especially for business in a rapid changing environment, like software development, people involved might not have enough time or do not want to spend much time searching for information or stored experience [3].

To address the above challenges, many researchers have turned to active task-oriented or process-oriented knowledge delivery approaches. In these approaches, knowledge delivery systems would be aware of specific task performed by users at specific time, and deliver necessary knowledge to accomplish that task to users. For that purpose, knowledge delivery system would "know" what information is needed for the task user is assigned to. This could be done during the enactment of business (or software) processes, in which, each task in the business process is clearly defined, as well as the necessary knowledge for such task. With the help of workflow management system, task-related knowledge can be delivered automatically to users [61].

There are two major research streams advocating this process-oriented knowledge management: project MILOS (Minimally Invasive Long-term Organizational Support) conducted at the University of Calgary and the University of Kaiserslautern; and project DÉCOR (Delivery of context-sensitive organisational knowledge) conducted at the German Research Center for Artificial Intelligence (DFKI).

MILOS [36] [37] focuses on KM needs in software project development. It aims at providing an infrastructure that integrates process-centred software engineering environments (PSEE) [23] with an Experience Factory KM approach proposed by [7]. In the MILOS system, knowledge is structured based on development processes. It is linked to processes that need to be carried out in the course of software development project. Such processes not only define the activities/tasks in the software project development, but also the knowledge needed to accomplish each task. Hence, when executing a task, a user will be “pointed” to the knowledge that is relevant for the task, in the form of links to related documents or people who have such knowledge. It can also get information from task context, thus providing more selective access to background knowledge, and reducing possible information overload.

Similar to project MILOS, project DÉCOR [2] [3] takes a business-process oriented knowledge management approach. In project DÉCOR, knowledge acquisition and delivery are highly integrated with business process management, powered by automatic workflow engine, so that context-specific knowledge is delivered to users when they were doing specific task. Compared to project MILOS, DÉCOR is more

focused on general business processes instead of software development processes. In such approach, not only fixed predefined information needs are attached to business tasks, but also the runtime information needs are parameterized by variables filled by the running workflow system, which can automatically pose a query to knowledge archives according to information needs. In this way, context-specific information retrieval can be performed which takes into account instance-specific information, thus getting more accurate results from knowledge archives.

The research on the process-oriented knowledge management leads to the third hypothesis:

***Hypothesis 3:** The more deeply knowledge delivery is embedded in software development process, the more effective is software development.*

3.4 Form of knowledge delivery: Inter-personal communication or reading document

Software development involves many document-driven processes and activities. Birk [9] illustrates a wide spectrum of software engineering processes that might occur in a typical software engineering project. He found that, among the results from all these processes and activities, what is common is that they are all documents. Documents are produced in different phases of the whole development life-cycle. They worked as

organization's knowledge repository to keep a variety of explicit knowledge. As software engineering is so dominated by documents, the foundation of KM systems in most software developing organization is Document Management Systems (DMS).

However, document cannot be taken as the only way for transferring knowledge. Learning from documents is not enough in software development organizations. Software engineers tend to talk directly with other people who have the knowledge they need, by discussing face to face, or via electronic means like telephone, e-mail in order to get more information for a better understanding of the knowledge.

As these two forms of learning knowledge: reading document or inter-personal communication, both exist in current software development practice, it is desirable to investigate how these two forms affect software development productivity.

Brown and Duguid [12] investigated the role played by documents in knowledge dissemination. They compared two basic models of document use: 'documents as darts' where documents serve as a means of objectively transmitting knowledge and 'documents as a means for supporting social processes'. They believed that the key role of documents in learning is to support negotiation and interpretation between communities as they (i.e. the communities) struggle to reach a shared understanding.

Moreover, software development is usually conducted in a volatile environment, characterized by fast and dynamic changes. So it is difficult to keep static documents updated with changes to deliver the latest information or knowledge. Therefore, we have:

Hypothesis 4: *Learning directly from people by inter-personal communication is more positively associated with software development productivity than reading documents.*

This chapter presents four key factors of knowledge delivery in software project development, as well as four hypotheses regarding the effect of these factors on software development productivity. Next chapter will focus on the research design – methods, variables (independent, dependent, and control), and the research model.

4 RESEARCH DESIGN

4.1 Research Method

4.1.1 Unit of Analysis

According to Pressman, metrics for measuring software development performance fall into three categories: process metrics, project metrics and product metrics [45]. Project metric is widely used to measure software development performance. In particular, software development productivity is used as the major metric to measure software development performance, as improving software development productivity is one of the most important goals of introducing KM in software development. It is also one of the key objectives of many software process improvement activities (Capability Maturity Model, Spice, Trillium, etc). In addition, most software development projects have the data used to measure project productivity, which makes it easy to be collected.

As project metrics will be used in the data analysis, the unit of analysis in this study is the software development project.

4.1.2 Sample selection

Companies involved in software development are considered as the target population for this study. This population includes companies of different sizes, who are

developing software product or combined software and hardware products for a variety of markets. These companies should have their headquarters in North America and with over 20 employees.

Key persons in the software development project teams in sampled organizations are targeted as respondents to answer the questionnaire. They were selected because of their familiarity with the sampled projects and because they are also responsible for project control.

Data used for hypothesis test in this study is collected through questionnaire. A questionnaire was sent to potential respondents by email. Completed questionnaires were also received via email, and then checked to make sure that they had been completed correctly.

Questionnaires were sent to 102 Canadian and American software development companies. The contact information of potential respondents in these companies was obtained from two sources: 1) technical papers or articles published in technical journal or conference; 2) personal networks. Using the first resource, the researcher contacted 47 authors of technical papers or articles that were published in year 2003. 6 of them agreed to participate in this research and have questionnaire answered. The response rate for this group is 12% (6/47). 55 potential respondents were contacted through personal networks, 37 of them answered the questionnaire and returned it to the researcher. For a total of 43

returned questionnaires, 41 of them are valid which have all questions answered. So the total response rate for this research is 40.2% (41/102).

The sample selection approach presented above does not strictly conformed to random selection. However, the respondents were selected in a way based on simulated random selection. The sample covers both large and small sized software development organizations, which are not confined to specific industry. In the sampled projects, 46% (or 19/41) are from large organizations that have more than 1000 employees, and 54% (or 22/41) are from small or medium-sized companies with less than 1000 employees.

4.1.3 Questionnaire

A two-page questionnaire was used to collect data used in this research, which is approved by Carleton University Research Ethics Committee in December 2003. The questionnaire was sent out to the potential respondent with a cover letter that is provided in Appendix A. The cover letter introduced the researcher and the supervisor for this study, as well as the detailed purpose of the research.

The questionnaire used in this research is provided in Appendix B. It includes three parts: 1) question 1-9 asked for the general information about the development project; 2) question 10-14 asked for how knowledge is delivered in such project; 3) question 15 –16 asked for the organization information.

The remaining part of this chapter describes all variables used in hypotheses testing and how to measure these variables, as well as the research model used in this research.

4.2 Variables and Measures

In this study, the independent variables are knowledge delivery factors presented in chapter 3. The dependent variables are software development productivity.

4.2.1 Independent Variables

- **How knowledge is delivered to receivers** is categorized by two approaches: *pull* approach and *push* approach. Respondents were asked to identify which one is the most often used approach for obtaining knowledge in their development projects.
- **Time of delivering knowledge** is measured by the time-frame between the time knowledge is delivered and the time when such knowledge is needed. Respondents were asked to use 7-point Likert scale to rate at what time they obtained work-related knowledge compared to the time when such knowledge is needed. “4” means the time of obtaining knowledge is same as the time such knowledge is needed. If the time of obtaining knowledge is earlier than the time of such knowledge is needed, respondents are expected to answer 1 – 3. Otherwise, if it is late, the expected answers are 5 - 7.

- **How knowledge delivery is embedded in software development process.** As described in both MILOS [36] and DÉCOR [2] projects, there are two major attributes of process-oriented knowledge delivery models, which will be used as indicators to measure how deeply knowledge delivery is embedded in software development process:

- If required knowledge for each task in the development process is clearly defined during the enactment of such process?
- In the run-time of the development process, when a knowledge worker is assigned to a specific task, will he/she easily access the necessary knowledge to complete such task?

A 7-point Likert scale that range from “totally disagree” (1) to “totally agree” (7) were used to obtain score for each attribute.

- **Form of knowledge delivery** falls into two categories: reading document and learning directly from other people via inter-personal communications. Respondents were asked to identify which one is the most often used form to obtain work-related knowledge in their project development.

4.2.2 Dependent Variables

Software development productivity is calculated by dividing software project size with development efforts:

$$Productivity = Software\ project\ size / Development\ Effort$$

In the current literature of software engineering, two metrics are most often used to measure software product size: *Line of Code* (LOC) and *Function Points* (FP) [30]. LOC is simple and easy to collect, but difference among programming languages will have huge variations in the counting of LOC. Function Point is the total number of functions provided in a given software product, in which case, function can be considered as a collection of executable statements of software that performs a certain task, together with declarations of the formal parameters and local variables manipulated by those statements [15]. In this research, FP is used as the metric to measure software project size, as it is independent of programming languages. It can provide more accurate results since it uses readily countable characteristics of software project that makes it easier to accommodate reuse and object-oriented approaches.

Participants in this research can indicate the size of their projects by either LOC or FP. In the data analysis, LOC will be converted to FP based on QSM Function Point Programming Languages Table [46]. As is widely used in software engineering literature, *Person Month* (Number of person involved in project multiply development time) will be used to measure software development efforts.

4.2.3 Control Variables

4.2.3.1 Organization size

Different type of companies may have different organization cultures, which will deeply affect how they manage knowledge. Compared with big companies, small and medium size companies have a very different way of learning and transferring knowledge because of limited resources and different business strategies. As a result, they would take different approaches or forms to deliver knowledge. Organization size will be measured by the total number of employees in the sampled company.

4.2.3.2 Knowledge Worker's Experience

To some extent, the effectiveness of various approaches for knowledge delivery (pull/push) is depended on the receivers' capabilities at identifying what knowledge is needed for their work at hand, and articulating the needed knowledge clearly to the knowledge delivery system. Furthermore, knowledge delivery will be more effective and faster if knowledge workers in the development team can quickly identify where or from whom to get necessary knowledge. These capabilities require much more skills and domain-specific knowledge for knowledge workers, which will be increased with more working experience accumulated. Therefore we believe that knowledge workers' experience will impact the effectiveness of knowledge delivery. Respondents were asked

to rate on a 7-point Likert scale the development team’s prior experience on developing such a product. The anchors are 1-none and 7-considerable.

4.3 Research Model

Figure 3 depicts the research model used in this research, which illustrates how different knowledge delivery factors affect software development productivity and how control variables impact knowledge delivery in software development as well.

Figure 3 - Research Model

5 RESEARCH RESULTS

5.1 Samples

As the distribution of the collected data determines what statistic method will be used for the data analysis, the researcher first applies descriptive statistics for numeric variables in the collected data to examine if they are normally distributed. A variable is considered to be normally distributed if its skewness was less than twice the standard error and its kurtosis was within the +2 and -2 range.

In this study, knowledge delivery approaches and knowledge delivery forms are measured on a nominal scale. Other six variables have numeric values. In particular, in order to minimize the effect of extreme values of software development productivity in the data analysis, natural logarithm of software development productivity is used as the dependent variable in the hypothesis testing. Unless otherwise mentioned, in the remaining part of this chapter, the term 'productivity' is the natural logarithm of software development productivity.

The SPSS software was used to provide descriptive statistics for these six variables, the results are presented in Table 1. Examination of Skewness and Kurtosis indicates that these variables are normally distributed, as their Skewness statistics are less than twice the standard error and the absolute value of their Kurtosis statistics are less than +2.

	N	Mean	Std.	Skewness		Kurtosis	
	Statistic	Statistic	Statistic	Statistic	Std. Error	Statistic	Std. Error
Knowledge Delivery Time	41	3.78	1.037	-.100	.369	-.709	.724
If knowledge is defined in process	41	4.68	1.312	-.002	.369	-.703	.724
If knowledge is easily accessed	41	5.32	1.234	-.311	.369	-.730	.724
Organization Size	41	2.88	1.503	.450	.369	-1.309	.724
Knowledge Worker's experience	41	5.10	1.357	-.059	.369	-.769	.724
Productivity	41	2.56309	.729706	-.158	.369	-.682	.724
Valid N (listwise)	41						

Table 1 - Descriptive Statistics (Sample Size = 41)

Table 2 presents the Pearson Correlations Coefficients for the numeric variables used in this study. The data provided in Table 2 indicates that organization size and knowledge worker's experience are strongly correlated with knowledge delivery time. The two independent variables used to measure embedded-ness of knowledge delivery in software development processes are strongly correlated with each other.

	Org Size	Exp	Knowledge Delivery Time (H2)	If knowledge is defined in process (H3)	If knowledge is easily accessed (H3)	Productivity
Org Size		.006	.479**	.246	.170	-.251
Experience			-.286*	.074	.160	-.086
Knowledge Delivery Time				.039	-.237	-.086
If knowledge is defined in process					.434**	-.301*
If knowledge is easily accessed						-.057

*p < 0.1, ** p < 0.05, *** p < 0.01

Table 2. Pearson Correlations Coefficients for Study Variables

5.2 Data analysis

As illustrated in Figure 3, knowledge delivery factors are independent variables in this research. The dependent variable is software development productivity. The major concern of the data analysis is to explore how different knowledge delivery factors affect software development productivity under different environments, which are indicated by two control variables: organization size and knowledge worker's experience.

In order to investigate how control variables impact the hypothesis testing, for each hypothesis, the researcher examines the interactions between control variables with independent variables in that hypothesis. If control variables and independent variables are strongly interacted, control variables will be taken into account when testing that hypothesis. Otherwise, the main effects of independent variables on software development productivity will be used to test that hypothesis.

Four hypotheses are proposed on the relationships of knowledge delivery factors with software development productivity. A null hypothesis is provided for each hypothesis, which is rejected or accepted based on the value of probability p from hypothesis testing. If $p > .10$, the null hypothesis is accepted and we believe that the examined independent variables do not have significant effects on software development productivity. If $p < .10$ the null hypothesis is rejected. Three levels of significances indicating the strength of impact of independent variables on software

development productivity will be used in the hypothesis testing: $p < .01$, $p < .05$, and $p < .10$.

Remaining parts of this chapter discusses in more details about the methods used to test the hypotheses, and present the testing results.

5.3 Hypothesis Testing

5.3.1 H1, Knowledge Delivery Approach

For knowledge delivery approach, respondents were asked to identify which approach, pull or push, was most often used for knowledge delivery in their projects. About two-third (27/41) of sampled projects are using pull approach for knowledge delivery, others are using push approach.

Sampled projects fall into two groups based on which approach they are taking for knowledge delivery. The effect of knowledge delivery approaches on software development productivity can be examined by comparing the average mean of productivity of these two groups of projects. Figure 4 depicts the box plots of software development productivity by projects using different approaches of knowledge delivery.

Figure 4 - Box plot of SWD Productivity by Knowledge Delivery Approaches

Table 3 presents the testing results of interactions between the two control variables and knowledge delivery approaches. Such results indicate that knowledge delivery approaches do not interact with either organization size or knowledge worker's experience. Therefore, H1 is tested without considering the effects of control variables.

	R^2 (adjusted)	F	Sig
Org-size * approach	0.130	1.715	0.183
Experience * approach	0.258	1.948	0.142

Table 3 - Interactions between Control Variables and Delivery Approaches

As the two knowledge delivery approaches are not numerically related to each other, *two independent-samples T test* is used to test H1. The null hypothesis (H_0) for H1 is that the population mean of software development productivity is the same for projects using the pull approach for knowledge delivery and projects using the push approach for knowledge delivery. The result of the two independent-samples T test of H1 is presented in Table 4.

	t-value	Sig. (2-tailed)	Mean difference	Mean	
				pull	push
Full Sample (n = 41)	2.628**	.012	.58034	2.77541	2.19507

*p < 0.1, ** p < 0.05, *** p < 0.01

Table 4 - Two Independent Samples t Test Results of H1

In Table 4, the “Sig. (2-sided)” column represents the significance of the comparison result: $p = 0.012$. This result indicates that the null hypothesis can be rejected with $p < 0.05$. As mean difference of productivity between projects using the pull approach and projects using the push approach is positive, we can conclude that projects using the pull approach for knowledge delivery are more productive than projects using the push approach for knowledge delivery. Therefore, H1 is accepted.

5.3.2 H2, Time of Knowledge Delivery

In H2, knowledge delivery time is measured by the time-frame between the time of knowledge delivery and the time when such knowledge is needed. As depicted in

Figure 5, respondents were asked to use Likert scale ranging from 1-7 to estimate the time of knowledge delivery in their projects. To state it more clearly, T_d is defined as knowledge delivery time; T_n is defined as the time when the required knowledge is needed. If the respondent answered 4 for the question about the knowledge delivery time, it means that T_d is same as T_n .

Figure 5 - Measurement of Knowledge Delivery Time

As believed in “Just-In-Time” knowledge delivery, knowledge delivery time should not be late or too earlier than the time such knowledge is needed. Using Figure 5 to describe that, it means that software development projects which have knowledge delivered “on-time” (score = 3 or 4) will have higher productivity than the projects with knowledge delivered too early (score < 3) or too late (score > 4).

To get more direct results for hypothesis testing, instead of using the value of T_d got from returned questionnaires, the absolute value of the difference between T_d and the ideal value of “on-time” knowledge delivery is used as the independent variable to test H2. As projects with T_d equalling to 3 or 4 are considered as projects with “on-time”

knowledge delivery, the median value (T_{ab}) between 3 and 4 (3.5) is used as the indicator for the ideal time of knowledge delivery, which is defined as $T_{ab} = |T_d - 3.5|$.

T_{ab} can be considered as the measurement of distance of T_d from the ideal value of knowledge delivery time. As a result, H2 can be tested by examine if T_{ab} is negatively correlated with software development productivity.

The impact of two control variables, organization size and experience, can be examined by testing the interactions between them and T_{ab} . The test results presented in Table 5 lead us to believe that T_{ab} does not interact with control variables.

	R^2 (adjusted)	F	Sig
Org-size * T_{ab}	0.192	1.900	0.149
Experience * T_{ab}	0.081	1.170	0.344

Table 5 - Interactions between Control Variables and Independent Variables for H2

Table 6 provides the descriptive statistics of T_{ab} . It suggested that T_{ab} is not conformed to normal distribution. Therefore, Spearman bi-variate correlation test is used to test H2. Testing results presented in Table 7 indicated that T_{ab} is significantly ($p < 0.05$) correlated with software development productivity at hypothesized sign, so H2 is accepted.

	N	Mean	Std.	Skewness		Kurtosis	
	Statistic	Statistic	Statistic	Statistic	Std. Error	Statistic	Std. Error
TIME-abs	41	.9146	.54661	.839	.369	-.370	.724
Valid N (listwise)	41						

Table 6. Descriptive Statistics of T_{ab}

			Productivity	Time
Spearman's rho	Productivity	Correlation Coefficient	1.000	-.370*
		Sig. (2-tailed)	.	.017
		N	41	41
	Time	Correlation Coefficient	-.370*	1.000
		Sig. (2-tailed)	.017	.
		N	41	41

*. Correlation is significant at the .05 level (2-tailed).

Table 7 - Spearman correlation of T_{ab} and SWD productivity

H2 can be further tested by the quadratic regression, which builds a non-linear model to describe the relationship between the time of knowledge delivery and software development productivity. The quadratic regression specification of H2 is that:

$$\text{SWD productivity} = \alpha + \beta_1(\text{time}) + \beta_2(\text{time})^2 + \varepsilon \text{ with } \beta_1 > 0 \text{ and } \beta_2 < 0,$$

Knowledge delivery time	R ² (adjusted)	F	Estimate of β_1	Estimate of β_2
SWD productivity	0.046	1.96	1.302417*	-0.182592*

*p < 0.1, ** p < 0.05, *** p < 0.01

Table 8 - Results of Quadratic Regression Analysis for H2

Table 8 presents the results of the quadratic regression analysis of H2. In such result, the estimate of β_1 and β_2 are significant and they have the hypothesized sign, which suggest that there is quadratic relationship between SWD productivity and knowledge delivery time. The best-fit quadratic equation for the regression – SWD productivity against knowledge delivery time – is:

$$\text{SWD productivity} = \text{constant} + 1.30(\text{time}) - 0.18(\text{time})^2$$

Based on this equation, software development productivity will reach the peak value when knowledge delivery time equals to 3.61 ($1.30/2*0.18$). The slope of equation is $1.30 - 0.18(2)(\text{time})$. When the knowledge delivery time at its maximum value of 7, this slope takes the value $1.30 - 0.18(2)(7) = -1.22$. The negative slope of the relationship between SWD productivity and knowledge delivery time at its maximum value supports H2.

5.3.3 H3, Embedded-ness of Knowledge Delivery in SWD Process

As mentioned in section 4.2.1, two independent variables are used to measure how deeply knowledge delivery is embedded in software development processes: a) if knowledge required for each task is clearly defined in the development process b) how easily knowledge worker can access the required knowledge at run-time. Score of each index is rated on 7-point Likert scale.

Table 9 presents the testing results of the interactions between control variables and these two independent variables. The results indicate that the independent variables used in H3 do not interact with the control variables. So H3 is tested by examining the main effects of independent variables on software development productivity.

	R ² (adjusted)	F	Sig
Org-size * knowledge define in process	-0.356	0.010	0.990
Experience * knowledge defined in process	-0.380	0.340	0.721
Org-size * knowledge is easily accessed	-0.356	1.414	0.301
Experience * knowledge is easily accessed	-0.380	0.773	0.490

Table 9 - Interactions between Control Variables and Independent Variables for H3

H3 aims to test if embedded-ness of knowledge delivery in software development processes is positively associated with software development productivity. As presented in Table 1, variables used to test H3 are conformed to normal distribution, *Person bi-variate ccorrelation* is used to test such relationship.

The results of such correlation testing are presented in part of Table 2. The results indicate that neither of the two independent variables used to measure embedded-ness of knowledge delivery in SWD processes is positively correlated with software development

productivity (Person correlation is -0.301 and -0.057 respectively). There is no evidence in the data to support H3.

5.3.4 H4, Knowledge Delivery Form

In the case of H4, respondents were asked to identify which form (learning directly from people through inter-personal communication or reading documents) is more often used for knowledge delivery in their projects. In about two-third (27/41) of the sampled projects, reading documents is the major form for knowledge delivery, one-third (14/41) of projects are using inter-personal communication as the major form of knowledge delivery.

Sampled projects fall into two groups based on which form they are taking for knowledge delivery: inter-personal communication or reading documents. How these two knowledge delivery forms affect software development productivity can be examined by comparing the average means of productivity of these two groups. Figure 6 depicts the box plots of software development productivity of these two groups of projects.

Figure 6 - Box plot of SWD Productivity by Knowledge Delivery Forms

The results of interactions between two control variables, organization size and experience, and of knowledge delivery forms are presented in Table 10. The results indicate that knowledge delivery forms do not interact with these two control variables. So H4 is tested without considering the effects of control variables.

	R ² (adjusted)	F	Sig
Org-size * form	0.012	1.588	0.211
Experience * form	-0.145	0.202	0.935

Table 10 - Interactions between Control Variables and Delivery Forms

Similar to H1, as the variables used to measure knowledge delivery forms are not numerically related to each other, *two independent-samples T test* is used to test H4. The null hypothesis (H_0) for H4 is that the population mean of software development productivity is same for projects using inter-personal communication as the form of knowledge delivery and projects using reading documents as the form of knowledge delivery. The results of two independent-samples t test results for H4 are provided in Table 11.

	t-value	Sig. (2-tailed)	Mean difference	Mean	
				communication	Reading document
Full Sample (n = 41)	0.059	.953	.01444	2.55358	2.56802

*p < 0.1, ** p < 0.05, *** p < 0.01

Table 11. Two Independent Samples t Test Results of H4

As shown in Table 11, there is no significant difference of software development productivity between projects with different knowledge delivery forms. Such results do not provide any evidence to support H4.

5.4 Summary

Table 12 summarizes the results of hypothesis testing for all hypotheses proposed in this research.

Hypothesis	Test Outcome	Rationale
Hypothesis H1. <i>Software development is more productive if majority of knowledge is delivered by pull approach rather than push approach.</i>	Supported	In Table 4, average mean of productivity of projects using pull approach is higher than the average mean of productivity of projects using push approach, with $t = 2.628$ at $p < 0.05$.
Hypothesis H2. <i>Software development productivity is impacted positively if knowledge required is delivered close to the time such knowledge is needed.</i>	Supported	In, Table 7, T_{ab} is negatively correlated with software development productivity at $p < 0.05$. In the quadratic regression model presented in Table 8, $\beta_2 < 0$ at $p < 0.1$. The slope of the relationship between knowledge delivery time and SWD productivity is negative when knowledge delivery time takes the maximum value.
Hypothesis H3. <i>The more deeply knowledge delivery is embedded in software development process, the more productive is software development.</i>	Not support	In Table 2, variables to measure embedded-ness of knowledge delivery in SWD process are negatively correlated with SWD productivity.
Hypothesis 4. <i>Learning directly from people by communication is more positively associated with software development productivity than reading documents.</i>	Not support	In Table 11, there is no significant difference of software development productivity between projects using different knowledge delivery forms.

Table 12 - Summary of Hypothesis Testing

6 DISCUSSION OF RESULTS

As stated in section 4.2, two control variables, organizational size and knowledge worker's experience, have deep influence on how knowledge is delivered in software development organizations. This chapter investigates the relationship between these two variables and knowledge delivery factors in software development. And then, based on the results of such investigation and hypothesis testing provided in chapter 5, discusses the effect of knowledge delivery factors on software development productivity one by one.

6.1 Influence of Control Variables on Knowledge Delivery

6.1.1 Organizational Size

Different size of companies may have different organization cultures and business strategies, which will influence how they manage their in-house knowledge. Therefore, how knowledge is delivered and shared in small/medium-sized companies may be very different from large-sized companies. The effect of software development organization's size on knowledge delivery in such organization can be examined by comparing the difference of the knowledge delivery factors between small/medium-sized (< 1000 employees) organizations and large-sized (>1000 employees) organizations.

In all returned questionnaires, 19 are from large organizations and 22 are from small/medium organizations. As all numeric variables in sampled data are normally

distributed, the following two approaches are taken to examine the impact of organizational size on knowledge delivery factors in software development:

- To examine the impact of organizational size on knowledge delivery approaches (pull or push) and knowledge delivery forms (inter-personal communication or reading document) Cross-Tabulation Pearson Chi-square test is used to compare the count of projects using different approaches or forms between small/medium-sized organizations and large-sized organizations. Test results are presented in Table 13.
- To examine the impact of organizational size on knowledge delivery time and embedded-ness of knowledge delivery in software development processes, two-tailed t test is applied to test the difference of the means of these two knowledge delivery factors between small/medium-sized and large-sized organizations. Test results are presented in Table 14.

Knowledge delivery factors		Large (n=19)	Small/Medium (n=22)	chi-Value
Approach	pull	10	17	2.753 *
	push	9	5	
Form	Communication	5	9	0.966
	Reading document	14	13	

*p < 0.1, ** p < 0.05, *** p < 0.01

Table 13 - Chi-Square Test Result for Organizational Size

Knowledge delivery factors	Large (n=19)		Small/Medium (n=22)		t-Value
	Mean	S.D	Mean	S.D	
Knowledge delivery time	4.26	.872	3.41	1.054	2.80 ***
Embedded-ness of Knowledge delivery in SWD process	5.105	1.1496	4.909	1.0309	0.576

*p < 0.1, ** p < 0.05, *** p < 0.01

Table 14 -Two Independent Samples T Test Results for Organizational Size

As shown in Table 13 and Table 14, compared with large-sized organizations, small/medium-sized organizations prefer to use the pull approach for knowledge delivery (*chi*-value = 2.753, $p < 0.1$). On average, small/medium-sized organizations have earlier knowledge delivery time than large-sized organizations (*t*-value = 2.80, $p < 0.01$). On the other hand, small/medium-sized organizations and large-sized organizations do not have marked difference on using which form, inter-personal communication or reading document, for knowledge delivery and on how knowledge delivery is embedded in SWD process.

6.1.2 Influence of Knowledge Worker's Experience

High-experienced software development teams may have different ways of delivering and sharing knowledge compared to low-experienced teams. The effect of knowledge worker's experience on how knowledge is delivered in software development

teams can be examined by comparing knowledge delivery factors between high-experienced teams and low-experienced teams.

Respondents were asked to rate on a 7-point Likert scale the project team's prior experience on developing similar products. Sampled projects are divided into two sub-groups based on the answers: low experience projects are projects with scores of 4 or less; high experience projects are those with scores of 5 or higher. Cross-Tabulation Chi-square test (for knowledge delivery approach) and two-tailed t test (for knowledge delivery time and embedded-ness of knowledge delivery in development process) are applied to compare the difference of knowledge delivery factors between the projects developed by high-experienced teams and the projects developed by low-experienced teams. Test results are presented in Table 15 and Table 16.

Knowledge delivery factors		Low Experience (n=15)	High Experience (n=26)	chi-Value
Approach	pull	10	17	0.007
	push	5	9	
Form	Communication	8	6	3.873**
	Reading document	7	20	

*p < 0.1, ** p < 0.05, *** p < 0.01

Table 15 - Chi-Square Test Result for Experience

Knowledge delivery factors	Low Experience (n=15)		High Experience (n=26)		t-Value
	Mean	S.D	Mean	S.D	
Knowledge delivery time	3.73	1.002	3.93	1.163	-0.588
Embedded-ness of Knowledge delivery in SWD process	4.981	1.0815	5.033	1.1095	-0.149**

*p < 0.1, ** p < 0.05, *** p < 0.01

Table 16 - Two Independent Samples T Test Results for Experience

As shown in Table 15 and Table 16, development teams with different experience do not have considerable difference on selecting which approach to take for knowledge delivery. However, high-experienced development teams are more likely to use reading document as the form of knowledge delivery (*chi*-value = 3.873, $p < 0.05$) than low-experience teams. Compared with high-experienced development teams, knowledge delivery in low-experienced teams is more deeply embedded in SWD processes (*t*-value = -0.149, $p < 0.05$).

6.2 Result Analysis for Knowledge Delivery Factors

– *Knowledge Delivery Approaches*

The result obtained from testing hypothesis H1 indicates that projects using pull approach for knowledge delivery will be more productive than projects using push approach for knowledge delivery. This leads us to believe that the software development

will be more productive if pull approach is used for knowledge delivery rather than push approach.

In addition, it is discovered that small/medium-sized organizations and high-experienced development teams prefer to use pull approach for knowledge delivery. This might be because small/medium-sized companies do not have enough resource to use push approach (like traditional class-room training) to deliver knowledge. Knowledge workers with more experience would probably know where and from whom to obtain the knowledge they needed, thus it would be easier and more effective for them to search for knowledge by themselves.

– *Knowledge Delivery Time*

The result obtained from testing hypothesis H2 suggests that software development projects with just-in-time knowledge delivery (knowledge is delivered close to the time such knowledge is needed) have higher productivity than projects in which knowledge is delivered too early or too late. Therefore, the researcher believes that the concept of just-in-time knowledge delivery advocated by Kevin Cole [14] would get more support from software development organizations.

– *Embedded-ness of Knowledge Delivery in Development Process*

This research did not find any evidence from testing hypothesis H3 that the embedded-ness of knowledge delivery in software development processes would affect software development productivity. In particular, the researcher found that knowledge delivery in low-experienced development teams is more deeply embedded in software development processes than high-experienced teams. Such result suggests that knowledge workers with low-experience might not have strong capability for searching necessary knowledge by themselves as high-experienced knowledge workers, thus they may have stronger needs for knowledge delivery system embedded in development processes (like MILOS [36] or DÉCOR [2]) to help them to obtain the knowledge they need.

– *Knowledge Delivery Forms*

The result of testing hypothesis H4 indicates that the two forms of knowledge delivery, reading documents or inter-personal communication, have the same impacts on software development productivity. About two-third of the sampled projects are using reading document as the major form of knowledge delivery, it reveals the fact that documentation is heavily used in software development for knowledge delivery.

Furthermore, the researcher discovered that high-experienced development teams prefer to use reading documents as the form of knowledge delivery, while low-experienced development teams tend to use inter-personal communications for knowledge delivery. This is probably because low-experienced knowledge workers may lack the ability of effectively using documentation to preserve and share knowledge in

the course of software project development, so they tend to use direct communication to deliver and share knowledge.

7 CONCLUSIONS, LIMITATIONS AND FUTURE WORK

7.1 Conclusions

This thesis is an empirical study that explored the relationship between knowledge delivery factors and software development productivity. The author identified four key knowledge delivery factors in software development and proposed four hypotheses on their impacts on software development productivity by reviewing research literatures on knowledge management and software engineering. The proposed hypotheses were tested against a sample obtained from 41 software development projects, which was collected through questionnaires returned from software development companies headquartered in North America.

The research results of this study suggest that:

- Software development projects using pull approach for knowledge delivery will be more productive than projects using push approach for knowledge delivery.
- Software development will have higher productivity if the required knowledge is delivered close to the time such knowledge is needed. Thus software development companies are more likely to advocate the so-called “Just-In-Time” knowledge delivery.

- Compared with large-sized companies, small/medium-sized software development companies prefer to use pull approach rather than push approach for knowledge delivery.
- Software development productivity is not affected by how deeply knowledge delivery is embedded in software development processes.
- Software development teams with low experience have stronger needs for deeper embedded-ness of knowledge delivery in software development processes.
- Two forms of knowledge delivery, inter-personal communication or reading documents, do not have significant difference in their impacts on software development productivity.
- Software development teams with high experience tend to use reading documents as the form of knowledge delivery, while low-experienced development teams tend to use inter-personal communication to share and deliver knowledge.

7.2 Limitations

This study has three major limitations. First of all, the sample data used in this research was collected from software development organizations headquartered in North America. Therefore, the result of this research may not apply to software development

organizations outside North America. A larger sample comprised of more software development companies in a wider scope would have been preferable.

Secondly, this research focuses only on the knowledge delivery, or knowledge distribution phase of the whole knowledge management life cycle presented in Figure 1. It does not cover how knowledge is created, preserved in software development, and their impacts on software development productivity. In addition, it does not investigate how characteristics of knowledge itself, like content of knowledge, or structure of knowledge, affect software development productivity.

Thirdly, in the research literature of knowledge management and new product development [40] [18], organizational and human factors have deep impacts on the effect of organization learning and knowledge transfer. This thesis investigated the effects of two organizational factors, organization size and knowledge worker's experience, on knowledge delivery in software development. Other factors, such as organization's culture, development team's structure, the attitude of management and development team members toward KMS are not covered in this research.

7.3 Suggestions for future research

The author has four suggestions for future studies.

First of all, this study is focused on the knowledge delivery phase of KM cycle. Additional research is needed to investigate how other two phases of KM life cycle, knowledge creation and knowledge integration phase affect software development performance.

Secondly, different characteristics of knowledge, like the content or the structure of knowledge, will have deep impacts on the effectiveness of knowledge delivery. More research works is needed to investigate how these characteristics affect knowledge delivery in software development, and to figure out which delivery method is better for a specific type of knowledge, thus getting knowledge delivery methods more tailored for different types of knowledge in software development.

Software development involves different types of knowledge workers, who play different roles in the course of software project development, like project managers, software designers, systems engineers. They have different expectations for how knowledge is delivered to them. Hence, the third suggestion for future research is to investigate if it helps to deliver knowledge in different formats or views to better meet these expectations from different knowledge workers in software development.

Lastly, software development organizations employing different development process models (waterfall, spiral, prototyping, etc.) might have different ways of sharing and delivering knowledge in software development. On the other hand, knowledge delivery, which is integrated in knowledge management models, is usually part of process

improvement activities in software development organizations. So it has deep impacts on software development processes as well. There is interaction between knowledge delivery and software development processes. More research is expected on the effect of such relationship on software development performance.

8 REFERENCES

1. Aamodt, A., and Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications* 7, 1994, pp 39-59.
2. Abecker, A, et al. (2001). Business-Process Oriented Delivery of Knowledge through Domain Ontology. Presented at the DEXA conference, TAKMA-2001, *Second International Workshop on Theory and Applications of Knowledge Management*, Munich, 3-7 September 2001.
3. Abecker, A and Mentzas, G.(2001). Active knowledge delivery in semi-structured administrative processes. In *KMGov-2001: 2nd Int'l Workshop on Knowledge Management in Electronic Government*, Siena, Italy, May 2001.
4. Abts, C., Boehm, B., and Bailey Clark, E. (2000) Observations on COTS Software Integration Effort Based on the COCOTS Calibration Database, *The 25th Annual NASA Goddard Software Engineering Workshop*, 2000.
5. Althoff, K.-D., Birk, A., Wangenheim, C. G. V., Tautz, C. (1998). CBR for experimental software engineering. In *Proc. Case-Based Reasoning Technology - From Foundations to Application*, 1998, pp. 235-254.
6. Althoff, K.-D., Bomarius, F. and Tautz, C. (1998). Using case-based reasoning technology to build learning software organizations. In *Proc. Interdisciplinary Workshop on Building, Maintaining, and Using Organisational Memories*, OM-98, 1998.

7. Basili, V. R., Caldiera, G., and Rombach, H. D. Experience Factory. In *Encyclopedia of Software Engineering (J. J. Marciniak, ed.)*, vol. 1, John Wiley Sons, 1994.
8. Basili, V. R., Caldiera, G., Mcgarry, F., Pajerski, R., Page, G. and S. Waligora, (1992). The Software Engineering Laboratory - An operational software experience factory, In *Proc. 14th Int. Conf. on Software Engineering (ICSE14)*, 1992, pp. 370-381.
9. Birk, A., Surmann, D., and Althoff, K.D. (1999). Applications of Knowledge Acquisition in Experimental Software Engineering. *11th European Workshop on Knowledge Acquisition, Modeling, and Management*, 1999, pp. 67-84.
10. Boehm, B. (1984). Software Engineering Economics, *IEEE Transactions on Software Engineering, Vol. 10, No. 1*, 1984, pp. 5-21.
11. Brössler, P. (1999). Knowledge management at a software house: An Experience report, In *Learning Software Organizations: Methodology and applications; Proc. 11th Int. Conf. on Software Engineering and Knowledge Engineering, SEKE '99, Kaiserslautern, Germany, June 16-19, 1999*, pp. 163-170.
12. Brown, J. S. and Duguid, P.(1996) The Social Life of Documents. *First Monday*, <http://www.firstmonday.dk/issues/issue1/documents/>, 1996, Date visited: Apr 08, 2004, date modified: Jun 05, 1996.
13. Chatters, B. (1999). Implementing an Experience Factory: Maintenance and evolution of the software and systems development process, In *Proc. IEEE Int. Conf. on Software Maintenance*, 1999, pp. 146-151.

14. Cole, K., Fischer, O., Saltzman, P.(1997). Just-in-time knowledge delivery. *Communications of the ACM*. Vol.40, Issue.7, July 1997. pp. 49 – 53.
15. Conte, S.D., Dunsmore, H.E., and Shen, V.Y. (1986). Software Engineering Metrics and Models, *The Benjamin/Cummings*, 1986.
16. Cooper, L.P.(2003). A research agenda to reduce risk in new product development through knowledge management: a practitioner perspective. *Journal of Engineering and Technology Management*. 20(2003) pp.117-140.
17. Cross, R. and Baird, L.(2000). Technology is not enough: improving performance by building organizational memory. *MIT Sloan Management Review*, Vol.41, No.3, Spring 2000, pp. 69-78.
18. Cummings, J.L. and Teng, B.S. (2003) Transferring R&D knowledge: the key factors affecting knowledge transfer success. *Journal of Engineering and Technology Management*, 20 (2003), pp.39–68.
19. Davenport, T.H., and Prusak, L.(1998). Working Knowledge, how organizations manage what they know. *Harvard Business School Press*, 1998.
20. Davenport, T.H., Long, D. W. D. and Beers, M. C.(1998). Successful knowledge management projects, *Sloan Management Review Winter*, 1998, pp. 43-57.
21. Fischer, G. and Otswald, J.(2001). Knowledge management: problems, promises, realities, and challenges. *Intelligent Systems*, IEEE, Vol.16, Issue.1, Jan/Feb 2001. pp. 60 –72.
22. Fouque, G. and Matwin, S.(1992). CAESAR: A System for Case Based Software Reuse, 7th *Knowledge-Based Software Engineering Conference, KBSE*, 1992, pp. 90-99.

23. Garg, P.K. and Jazayeri, M.(1996). Process-centered Software Engineering Environments. *IEEE Computer Society Press*, 1996, pp. 119-130.
24. Gibson, M. and Kevin, C (1995). Domain Knowledge Reuse During Requirements Engineering, *7th International Conference on Advanced Information Systems Engineering, CAISE*, 1995, pp. 283-296.
25. Henninger, S. (1995). Information Access Tools for Software Reuse. *Journal of Systems and Software, Vol. 30, No. 3*, 1995, pp. 231-247.
26. Hoch, D. J., Roeding, C. R., Purkert, G. and Lindner, S. K. (2000). Secretes of Software Success: Management Insights from 100 Software Firms around the World, *Boston, Massachusetts, Harvard Business School Press*, 2000.
27. Houdek, F., Schneider, K. and Wieser, E. (1998). Establishing experience factories at Daimler-Benz. An experience report, In *Proc. 20th Int. Conf. on Software Engineering (ICSE20)*, 1998, pp. 443-447.
28. Johansson, C., Hall, P. and M. Coquard. (1999). "Talk to Paula and Peter - They are experienced" - The experience engine in a nutshell, In *Learning Software Organizations: Methodology and Applications; Proc. 11th Int. Conf. on Software Engineering and Knowledge Engineering, SEKE '99, Kaiserslautern, Germany, June 16-19*, 1999, pp. 171-186.
29. Jorgensen, M., Conradi, R. and D. Sjoberg. (1998), Reuse of software development experience at Telenor Telecom Software, In *Proc. European Software Process Improvement Conference (EuroSPI098)*, 1998.
30. Kan, S. H. (2003). Metrics and Models in Software Quality Engineering, 2nd Edition. *Addison-Wesley*, Chap 4.

31. Landes, D., Schneider, K. and Houdek, F., (1999), Organizational learning and experience documentation in industrial software projects, *Int. J. Human-Computer Studies* 51,1999, pp. 643-661.
32. Leenders, A.J., et.al. (2003). Virtuality, communication, and new product team creativity: a social network perspective, *Journal of Engineering and Technology Management*. 20(2003) pp. 69–92.
33. Mahé, S. and Rieu, C.(1997) Towards a Pull-Approach of KM for Improving Enterprise Flexibility Responsiveness: A Necessary First Step for Introducing Knowledge Management in Small and Medium Enterprises. In *ISMICK '97: Proceedings of the International Symposium on Management of Industrial and Corporate Knowledge* Compiegne, 1997.
34. Mahé, S. and Rieu, C. (1998). A pull approach to knowledge management: Using IS as a knowledge indicator to help people know when to look for knowledge reuse, In *PAKM-98: Practical Aspects of Knowledge Management. Proc. of the Second Int. Conference*. Oct. 1998.
35. Mark, W. (1992). Comet, *7th Knowledge-Based Software Engineering Conference, KBSE*, 1992, pp. 254-255.
36. Maurer, F. and Dellen, B.(1998). An Internet Based Software Process Management Environment, In *ICSE 98 Workshop on Software Engineering over the Internet*. 1998.
37. Maurer, F., Holz, H.(1999). Process-oriented knowledge management for learning software organizations. *Proceedings of 12th Knowledge Acquisition For Knowledge-Based Systems Workshop 1999 (KAW'99)*, Banff, Canada, 1999.

38. McGarry, F., et.al. (1994), Software Process Improvement in the NASA Software Engineering Laboratory, *CMU/SEI-95-TR-22*, Department of Computer Science, University of Maryland, College Park, MD 20742, 1994.
39. Merriam-Webster, O.(2001). “The Language Centre, Merriam-Webster's Online Dictionary & Thesaurus”, <http://www.m-w.com/>, 2001, date visited: Apr 08, 2004.
40. Mohrman, S.A., et.al.(2003). An empirical model of the organization knowledge system in new product development firms, *Journal of Engineering and Technology Management*. 20 (2003) pp. 7–38
41. Nonaka, I. And Takeuchi, H.(1995). “The Knowledge Creating Company”, *Oxford University Press*, 1995, p8
42. O'Leary, D.E.(1998). Knowledge-Management Systems: Converting and Connecting. *Intelligent Systems, IEEE*, Vol. 13, Issue. 3. May/June 1998, pp. 30 – 33.
43. O'Sullivan, A.(2003). Dispersed collaboration in a multi-firm, multi-team product-development project, *Journal of Engineering and Technology Management*. 20(2003) pp.93-116
44. Perfect consortium, PIA Experience Factory, The PEF Model, *ESPRIT Project 9090 D-BL-PEF-2-PERFECT9090*, 1996, available at <http://www.iese.fraunhofer.de/perfect/ Handbook/D-BL-Launch.pdf>, date visited: Apr 08, 2004.
45. Pressman, R. S. (1997). Software Engineering: A Practitioner's Approach, 4th Edition. *McGraw Hill*, Chap 4.

46. QSM Function Point Programming Languages Table. (2002).
<http://www.qsm.com/FPGearing.html>, date visited: Mar 13, 2004, date modified:
Jul, 2002.
47. Rombach, H.d., Basili, V.R. and Selby, R.W, editors (1992). Experimental
Software Engineering Issues: Critical Assessment and Future Directions.
Proceedings. *Lecture Notes in Computer Science No. 706*, Springer-Verlag, 1992.
48. Rus, I. and Lindvall, M.(2002). Knowledge Management in Software Engineering.
IEEE Software, Vol. 19, Issue. 3. May/June 2002, pp.26-38.
49. Schoen, E. (1991). Active Assistance for Domain Modeling, In *Proceedings of 6th
Knowledge-Based Software Engineering Conference, KBSE*, 1991, pp. 26-35.
50. Seaman, C., Mendonca, M., Basili, V., and Kim, Y.-M. (1999), An experience
management system for a software consulting organisation, In *Proc. Twenty-
Fourth Annual NASA Software Engineering Workshop*, 1999.
51. Smith, D. (1991). KIDS: A Knowledge-Based Software Development System,
Automating Software Design, MIT Press, 1991, pp. 483-514
52. Smith, T. and Setliff, D.(1992). Knowledge-Based Constraint-Driven Software
Synthesis, *7th Knowledge-Based Software Engineering Conference, KBSE*, 1992,
pp. 18-27
53. Sumner, T., et.al. (1998). Enriching Representations of Work to Support
Organisational Learning. [http://kmi.open.ac.uk/projects/enrich/enrich-oms98-
paper.html](http://kmi.open.ac.uk/projects/enrich/enrich-oms98-paper.html), date visited: Apr 04, 2004.
54. Sumner, T., Domingue, J., Zdrahal, Z., Millican, A., Murray, J.,et.al.(1999).
Moving from On-the-Job Training towards Organisational Learning. *Proceedings*

- of the 12th Banff Knowledge Acquisition Workshop, Banff, Alberta, Canada, Oct.1999, pp.16-22
55. Szulanski, G.(1996). Exploring internal stickiness: impediments to the transfer of best practice within the firm. *Strategic Management Journal*, Vol.17, Winter 1996, pp.27–43.
 56. Wiig, K. M. (1997). Knowledge management: Where did it come from and where will it go? *Expert Systems with Applications* 13, 1997. pp.1-14.
 57. Wiig, K. M. (1995). Knowledge Management Methods, *Schema Press*, 1995, p. 489.
 58. Yankee Group (2000). *The Economist*, Nov 18, 2000
 59. Ye, Y. (2001). An Active and Adaptive Reuse Repository System, *CD – ROM [Electronic Edition]*, 34th Hawaii International Conference on System Sciences, Software Technology Track, 2001
 60. Zack, M.H.(1999). Developing a knowledge strategy. *California Management Review*, Vol. 41, No.3, Spring 1999, pp. 125-145.
 61. Zhao, J. L. (1998). Knowledge management and organizational learning in workflow systems. In *Proceedings of AIS 98*, Baltimore, Aug, 1998, pp. 14-16

APPENDIX A Letter of Information

Date: 12/02/2003

Dear Sir/Madam:

My name is Zheng Sun. I'm a graduate student in Telecommunication and Technology Management (TTM), Department of Systems and Computer Engineering, Carleton University, Canada, supervised by Dr. Samuel A. Ajila. I'm doing a research study to identify 'how knowledge is delivered to software engineers in software project development', and investigate how different factors in knowledge delivery affect software development performance. I sincerely invite you to participate in my research study by answering questions listed in the attached questionnaire.

This questionnaire has been designed so you could complete and return it electronically by e-mail. Once you have completed the questionnaire, save your answers and attach the file as a reply to the e-mail sent to you originally. To complete the questionnaire, please select a completed software development project in which you were involved in the past three years. If the project involved more than developing a software product (e.g. developing a hardware product), please focus on the software development aspects of the project and the team responsible for developing the software product.

This questionnaire should take less than 30 minutes to complete. Most of questions are focused on how software engineers in specific software development project in your company obtain work-related knowledge. You're entitled to decline answering any question(s) in such questionnaire.

There are no anticipated risks to you for participating in this research study. You are not required to provide your names or other personal information to us and I'll delete any identifier in your email to protect your anonymity. However, if you allow us to

contact you when additional information is required, please provide your contact information at the end of questionnaire.

The information you provide will be kept strictly confidential. Only researchers and his supervisor can access such data. That data will be saved in researcher's laptop until the time when such research is complete, which is expected to be in June 2004. After that, the data will be deleted permanently.

Due to the anonymous nature of the survey, it is not possible to withdraw your information from other survey respondents. The time and efforts you take to participate in this research study is appreciated, but you may not benefit from your participation in the study. After the study, a copy of research results will be sent to you upon request by e-mail.

Our research results will be presented in 13th International Conference on Management of Technology (IAMOT) held in Washington, D.C, April, 2004. They'll also be presented to faculties and students in TTM program of System and Computer Engineering Department in Carleton University.

Should you have any questions about this research study or wish to withdraw from it, feel free to contact the researcher at +1 613 301-9185, or zsun2@connect.carleton.ca by e-mail. My supervisor, Dr. Samuel A. Ajila, can be reached at +1 613 520-2600 Ext. 2673, or Samuel.Ajila@sce.carleton.ca by e-mail.

This research has been reviewed and received ethics clearance through the Carleton University Research Ethics Committee. Should you have any concerns or questions about your involvement in this research study, please contact:

Carleton University Research Ethics Committee
511A Tory Building
Carleton University

1125 Colonel By Drive
Ottawa, ON K1S 5B6
Tel: +1 613 520-2517

Thanks very much for your time and co-operation.

Zheng (Bill) Sun
zsun2@connect.carleton.ca
Tel: +1 613 301-9185

Supervisor: Dr. Samuel A. Ajila
Samuel.Ajila@sce.carleton.ca
Tel: +1 613 520-2600 Ext. 2673

APPENDIX B Questionnaire

1125, Colonel By Drive, Ottawa, Ontario, K1S 5B6, Canada

Objective

This questionnaire is used to examine the effects of knowledge delivery factors of KM system on software development effectiveness.

Introduction

This questionnaire has been designed so you could complete and return it electronically. Once you have completed the questionnaire, save your answers and attach the file as a reply to the e-mail sent to you originally.

Please select a completed software development project with which you were involved in the past three years and answer the following questions. If the project involved more than developing a software product (e.g. developing a hardware product), please focus on the software development aspects of the project and the team responsible for developing the software product.

Questions about the survey process or content should be directed to: zsun2@connect.carleton.ca.

ALL RESPONSES WILL BE KEPT CONFIDENTIAL. The research results will be sent to you once they become available.

Thank you for collaborating in this research effort.

1. Name of project: _____

2. Your role in the project: _____

3. Please provide a brief description of the project (optional)

4. Please indicate what type of programming language used in coding the system

- ☐ Java
- ☐ C++
- ☐ 4th Generation Script Language
- ☐ Other

5. Please indicate the size of the project: _____ in Lines of Code (LOC) _____
OR in Function Points (FP) _____

If LOC is used and in case of more than one programming languages, could you please separate the lines of code (LOC) based on each programming language:

Java _____ C++ _____ Script language _____ Other _____

6. If possible, please indicate the total efforts consumed in that project in person-months:

_____ p/m.

(**Note:** you may skip question 7, 8 if you answer question 6)

7. Please indicate total size of team for the entire project: _____ persons

8. Please indicate the total number of people in the project team spending 100% of the time on the project: _____ persons. If applicable, please indicate the percentage of time spent on the project by the rest of the team who are not fully involved in the project:

Choose ONE (place an X in the appropriate bracket)

- ☐ Up to 80%
- ☐ Up to 60%
- ☐ Up to 40%
- ☐ Up to 20%

9. Please indicate the start date of the project (i.e. the first date when personnel were first assigned to the project in order to analyse the requirements):

_____/_____/_____ (dd/mm/yyyy)

Please indicate the end of project data (i.e. the date the project was released and install for the user): _____/_____/_____ (dd/mm/yyyy)

Please indicate if there is a slack time in the project duration (i.e. period of time(s) when no one was working on the project): _____ (months)

10. Please compare two approaches described in choice a) and b) and choose which approach is more often used in your development team to obtain work-related knowledge.

Choose ONE option (place an X in the appropriate bracket)

- ☐ a) Knowledge is pushed to team members. Team members obtain knowledge passively, by taking training or seminars, from newsletters or mailing list, or from automatic help system included in IDE (integrated development environment).
- ☐ b) Team members have to search for knowledge by themselves, pulling knowledge from organization's existing knowledge repository, from Internet through search engines, or by asking questions to other experts.

11. On a scale of 1 to 7 (1 being long before the knowledge is needed and 7 being long after the knowledge is needed), please indicate, in most cases, what time did your team members obtain work-related knowledge compared with the time when such knowledge is needed.

Choose ONE options (place an X in the appropriate bracket)

- ☐ a) Long before the time when such knowledge is needed
- ☐ b)
- ☐ c)
- ☐ d) almost at the same time when such knowledge is needed
- ☐ e)
- ☐ f)
- ☐ g) Long time after the time when such knowledge is needed

12. On a scale of 1 to 7, please indicate, if the process model followed by your development team clearly defines what knowledge is required to accomplish each task or activity in that process?

Choose ONE options (place an X in the appropriate bracket)

- ☐ a) required knowledge to accomplish each task is not defined at all
- ☐ b)
- ☐ c)
- ☐ d) almost 50% of such knowledge is defined
- ☐ e)
- ☐ f)

☐ g) all required knowledge to accomplish each task is defined.

13. On a scale of 1 to 7, please indicate, when team member is assigned to a specific task, will he/she be able to clearly understand what knowledge is needed to accomplish such task, and easily access the necessary knowledge to accomplish such task?

Choose ONE (place an X in the appropriate bracket)

- ☐ a) - totally disagree
- ☐ b)
- ☐ c)
- ☐ d) - somewhat agree
- ☐ e)
- ☐ f)
- ☐ g) - totally agree

14. Please compare two forms of learning knowledge as described in choice a) and b) and choose which form is more often used in your development team to obtain work-related knowledge.

Choose ONE (place an X in the appropriate bracket)

- ☐ By reading document in all formats (paper & electronic media)
- ☐ By learning directly from other people via communication (include face-to-face discussion, discussion via electronic means like telephone, email, video-conference)

15. Please indicate the size of your company by number of employees?

Choose ONE (place an X in the appropriate bracket)

- ☐ a) < 100
- ☐ b) 100 - 1000
- ☐ c) 1000 - 5000
- ☐ d) 5000 - 10000
- ☐ e) > 10000

16. Please rate the overall project team's prior experience in developing such a product?

Choose ONE (place an X in the appropriate bracket)

- ☐ a) - None
- ☐ b)
- ☐ c)
- ☐ d) - Some
- ☐ e)
- ☐ f)
- ☐ g) - Considerable

The information below is not mandatory, but would allow us to thank you and contact you should additional information be required.

Name:

Position:

Your Business Telephone Number:

Your Business Address: